

(19) 日本国特許庁 (J P)

## (12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-242286

(43) 公開日 平成8年(1996)9月17日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
H 0 4 M 3/00			H 0 4 M 3/00	D
3/22			3/22	Z

審査請求 未請求 請求項の数7 O L (全 18 頁)

(21) 出願番号 特願平7-44569

(22) 出願日 平成7年(1995)3月3日

特許法第30条第1項適用申請有り 1994年9月26日～9月29日 社団法人電子情報通信学会主催の「電子情報通信学会1994年秋季大会－ソサイエティ先行大会－」において文書をもって発表

(71) 出願人 000005223

富士通株式会社

神奈川県川崎市中原区上小田中4丁目1番1号

(72) 発明者 浜田 健生

神奈川県川崎市中原区上小田中1015番地  
富士通株式会社内

(72) 発明者 伊勢田 衡平

神奈川県川崎市中原区上小田中1015番地  
富士通株式会社内

(74) 代理人 弁理士 伊東 忠彦

最終頁に続く

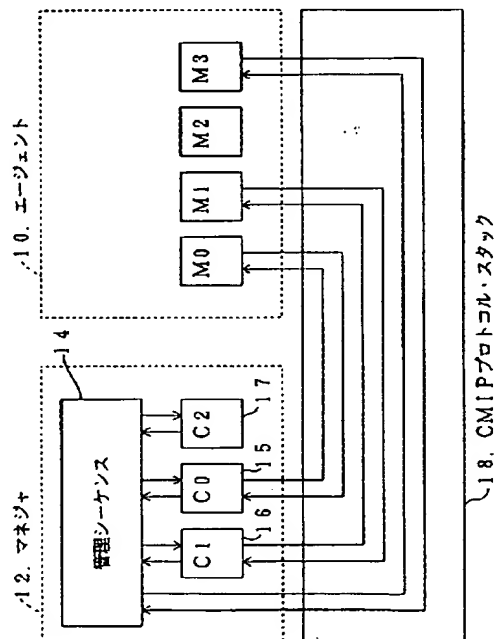
## (54) 【発明の名称】 通信網管理制御方式

## (57) 【要約】

【目的】 本発明は通信管理制御方式に関し、マネジャが実行する管理シーケンスを高速化し、キャッシュ管理を最適化してキャッシュ・コヒーレンスを確保すると共に管理トラヒックの増大を防止し、キャッシュをエージェントの障害復旧に有効利用することを目的とする。

【構成】 通信ネットワークの管理主体12から代理主体10にコマンドを送り、代理主体で通信ネットワークの管理情報のうちの管理対象を操作する通信網管理制御方式において、管理主体上に管理情報を格納するキャッシュ15～17を設け、キャッシュを管理情報のキャッシュ・コヒーレンス要求度に対応した属性クラスに基づいて分割する。

本発明の構成図



## 【特許請求の範囲】

【請求項 1】 通信ネットワークの管理主体から代理主体にコマンドを送り、上記代理主体で通信ネットワークの管理情報のうちの管理対象を操作する通信網管理制御方式において、

上記管理主体上に管理情報を格納するキャッシュを設け、

上記管理情報の属性をキャッシュ・コヒーレンシ要求度に対応してクラス分けしたことを特徴とする通信網管理制御方式。

【請求項 2】 前記管理情報の定義から前記属性クラスを得て、属性クラス毎のキャッシュ領域の確保及び初期値の設定を実行する初期化手段を自動生成することを特徴とする請求項 1 記載の通信網管理制御方式。

【請求項 3】 前記管理情報の定義から前記属性クラスを得て、属性クラス毎のコヒーレンシ・プロトコルを実行するインタフェース処理手段を自動生成することを特徴とする請求項 1 又は 2 記載の通信網管理制御方式。

【請求項 4】 前記属性クラスに基づいて分割されたキャッシュを上記属性クラスに基づいて管理し、かつ属性クラスに対応したコヒーレンシ・プロトコルを適用してキャッシュ・コヒーレンシを確保することを特徴とする請求項 1 乃至 3 のいずれかに記載の通信網管理制御方式。

【請求項 5】 前記属性クラスに基づいて分割されたキャッシュ夫々をリスト・リーセントリー・ユースト・アルゴリズムにより管理することを特徴とする請求項 1 乃至 4 のいずれかに記載の通信網管理制御方式。

【請求項 6】 前記代理主体に、自らと接続されている管理主体の情報を記録する不揮発性記録手段を有することを特徴とする請求項 1 記載の通信網管理制御方式。

【請求項 7】 前記代理主体の障害復帰時に、前記不揮発性記録手段に記録された管理主体上のキャッシュから上記代理主体の管理情報を復旧することを特徴とする請求項 6 記載の通信網管理制御方式。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】 本発明は通信網管理制御方式に関し、オブジェクト指向のネットワークの通信網管理制御方式に関する。最近、オブジェクト指向の考えで統一される TMN (Telecommunication Management Network) と呼ばれる通信網管理制御方式が開発されている。

【0002】 通信の広帯域化、高信頼化への要求は通信網のネットワーク管理の発展を促し、ITU-T (CCITT) においてもネットワーク管理の国際標準として TMN に関する勧告 M. 3100 が行われている。高信頼かつ高効率なネットワーク管理を行う為には、網管理に要する処理上の負荷、通信上の遅延に対処するような通信網管理制御方式が要求されている。

## 【0003】

【従来の技術】 TMN に基づく通信網管理では、通信機器の状態や通信網の接続状態等の管理情報は MO (Managed Object) と呼ばれるオブジェクトとして表現される。通信管理制御は、マネジャ (Manager) と呼ばれる管理主体が CMIP (Common Managed Information Protocol) と呼ばれる通信プロトコルを用いて、エージェント (Agent) と呼ばれる代理主体にコマンドを送り、エージェントはそのコマンドの内容に従って管理対象の MO を操作することによって行われる。

10 【0004】 エージェント上での MO の集合は管理情報ベース (MIB) と呼ばれる。また、MO の属性 (Attribute) は管理情報を構成する要素であり、例えばスイッチのオン/オフ状態等の物理的な意味を持っている。例えば、マネジャはエージェントに対して CMIP のコマンドを送ることにより、MO の属性値を読み出し (MGET)、又は書き込む (MSET)。通信網管理は通常このような CMIP コマンドのシーケンス (管理シーケンス) によって行われる。

## 【0005】

20 【発明が解決しようとする課題】 従来、TMN による通信管理を大規模なネットワークに適用した場合、マネジャとエージェントとの間の網内遅延が大きくなり、この網内遅延に CMIP のプロトコル処理に要する遅延を加えると、遅延時間が数十 msec に達する場合があった。このため、トラヒック状況や異常監視等のために、マネジャがエージェント上の MO 属性情報に頻繁にアクセスする必要のある管理シーケンスは実行に大きな時間がかかり、高速化が困難であるという問題があった。

30 【0006】 一方、キャッシュを用いて網内遅延を小さくする手法が以前から知られている。キャッシュを用いることにより、アクセス時間、特に属性値の読み出しに要する時間を大幅に短縮できる。しかし、キャッシュを用いる場合、各キャッシュ間の内容の一致 (コヒーレンシ) を保証する必要がある、コヒーレンシ・プロトコルを用いて常にコヒーレンシを保つ必要があった。そのためにキャッシュの数の増加に伴ってコヒーレンシ・プロトコルによるメッセージが増加し、このことにより管理トラヒックが増大しネットワークの負荷が増え、通信網管理の運用方法によってはネットワークが輻輳状態に陥る危険性があった。つまり、従来の技術ではキャッシュを持つ事の利点と、ネットワークの管理トラヒックの増大を抑えるという目標を両立させることが困難であり、このため通信網管理においてキャッシュ技術は広く用いられるに到らなかった。

40 【0007】 また、エージェントに障害が起こった場合に対処するため、MO の属性値をデータベースに格納し、障害復旧時にデータベース上の値を用いて MO の属性値を復旧する方法がとられる。また、同じ目的で MO の属性値の変更時に複数のマシンにコピーすることによって MO 属性値の保存を可能にする方法が知られてい

る。より具体的には、複数のホスト上にデータベースを配置して、一箇所のデータベースでMO属性値に変更があれば、他のデータベースにも同じ値をコピーする。この方法によれば、障害時のフォールト・トラレンシーは向上するが、コピーはマネジャ上のキャッシュとは独立であり、マネジャ上の管理アプリケーションによりキャッシュとして用いられることはなく、このため、通信網管理システムの高速化には直接貢献せず、またキャッシュを独立に設けた場合は同じ情報をデータベースに重複して持つことになり、効率が悪いという問題があった。

【0008】本発明は上記の点に鑑みなされたもので、マネジャが実行する管理シーケンスを高速化し、キャッシュ管理を最適化してキャッシュ・コヒーレンシを確保すると共に管理トラヒックの増大を防止し、キャッシュをエージェントの障害復旧に有効利用する通信管理制御方式を提供することを目的とする。

【0009】

【課題を解決するための手段】請求項1に記載の発明は、通信ネットワークの管理主体から代理主体にコマンドを送り、上記代理主体で通信ネットワークの管理情報のうちの管理対象を操作する通信網管理制御方式において、上記管理主体上に管理情報を格納するキャッシュを設け、上記管理情報の属性をキャッシュ・コヒーレンシ要求度に対応してクラス分けする。

【0010】請求項2に記載の発明は、前記管理情報の定義から前記属性クラスを得て、属性クラス毎のキャッシュ領域の確保及び初期値の設定を実行する初期化手段を自動生成する。請求項3に記載の発明は、前記管理情報の定義から前記属性クラスを得て、属性クラス毎のコヒーレンシ・プロトコルを実行するインタフェース処理手段を自動生成する。

【0011】請求項4に記載の発明は、前記属性クラスに基づいて分割されたキャッシュを上記属性クラスに基づいて管理し、かつ属性クラスに対応したコヒーレンシ・プロトコルを適用してキャッシュ・コヒーレンシを確保する。請求項5に記載の発明は、前記属性クラスに基づいて分割されたキャッシュ夫々をリスト・リーセントリー・ユースト・アルゴリズムにより管理する。

【0012】請求項6に記載の発明は、前記代理主体に、自らと接続されている管理主体の情報を記録する不揮発性記録手段を有する。請求項7に記載の発明は、前記代理主体の障害復旧時に、前記不揮発性記録手段に記録された管理主体上のキャッシュから上記代理主体の管理情報を復旧する。

【0013】

【作用】請求項1に記載の発明においては、管理主体にキャッシュを設けることにより、管理主体が実行する管理シーケンスを高速化することができ、管理情報の属性をキャッシュ・コヒーレンシ要求度に基づいてクラス分けすることによりキャッシュ管理の最適化をはかること

が可能となる。

【0014】請求項2に記載の発明においては、管理情報の定義から初期化手段が自動生成され、この初期化手段によって管理主体に属性分割キャッシュが確保され、かつ、その初期化が行われ、属性分割キャッシュを使用しない装置との相互運用性を確保できる。

【0015】請求項3に記載の発明においては、管理情報の定義からインタフェース処理手段が自動生成され、このインタフェース処理手段によって属性クラス毎のコヒーレンシ・プロトコルが実行されるため、属性キャッシュ夫々に最適のコヒーレンシ・プロトコルが実行可能となる。

【0016】請求項4に記載の発明においては、属性分割キャッシュが夫々に最適のコヒーレンシ・プロトコルを用いてキャッシュ・コヒーレンシが確保されるため、ネットワークの管理トラヒックの増大を防止してネットワークの負荷の増大を抑制できる。

【0017】請求項5に記載の発明においては、属性分割キャッシュをリスト・リーセントリー・ユースト・アルゴリズムを用いて管理するため、キャッシュ領域が増大化することを避けることができ、メモリの有効利用が可能となる。請求項6に記載の発明においては、不揮発性記録手段を有するため、代理主体と接続されている管理主体の情報を代理主体の障害時に確保できる。

【0018】請求項7に記載の発明においては、代理主体の障害復旧時に、これと接続されていた管理主体のキャッシュから管理情報を復旧できるため、キャッシュを代理主体の障害復旧に有効利用できる。

【0019】

【実施例】図1は本発明の構成図を示す。同図中、エージェント10は管理情報属性を物理的性質やキャッシュ・コヒーレンシの要求度の高さにより分割し、属性クラスA～Dの管理情報M0～M3に分けて管理している。

【0020】マネジャ12は管理シーケンス14を有すると共に、キャッシュ15、16、17を有している。キャッシュ15～17はエージェント10における属性クラスA～Cの管理情報M0、M1、M2夫々に対応して設けられており、属性クラスA～Cの管理情報C0、C1、C2を格納する。CMIPプロトコル・スタックは上記エージェント10とマネジャ12との間を接続しており、両者間でCMIPコマンドが送受されてエージェント10の管理情報がマネジャ14に送られる。

【0021】上記の属性クラスA～Dについて説明する。属性クラスAは、一時的なキャッシュの不一致が許される属性であり、CMIPコマンドの“GET”でマネジャから読み出し操作のみが許されるGET-ONLY属性である。この属性としては監視情報のMO、パスに関するコスト情報のMO、網の統計情報のMO等がある。

【0022】属性クラスBは、厳密なコヒーレンシが必

要な属性であり、CMIPコマンドの“GET-REPLACE”でマネジャから読み出し又は書き換え操作により属性値の設定が可能なGET-REPLACE属性である。この属性としては、ドメイン境界のクロスコネクト情報のMO等がある。

【0023】属性クラスCは、GET-ONLY属性であり、CMIPコマンドのM-CREATEを用いたMOの生成時に固定され、MOの存在期間において変化する可能性のない固定的な属性である。属性クラスDは、一旦生成されると読み出されることは殆どなく、キャッシュを設ける必要性の無い属性である。

【0024】ある属性がどの属性クラスに属するかの分割は、図2に示す如きGDMO定義に与えられる類別(GET, GET-REPLACE)から大まかに可能であり、その属性の物理的な意味からどのクラスに属するかを判別できる。図2においては、ステップS100, S110からoperationalStateGET-ONLY属性であり、物理的な意味から属性クラスAである。ステップS80, S90からadministrativeStateがGET-REPLACE属性であるので属性クラスBである。ステップS60, S70, S120~S190からcrossConnectionID, signalType, from Termination, to Termination, directionality夫々がGET-ONLY属性であり、物理的な意味から属性クラスCである。図2の例では属性クラスDは存在しない。

【0025】上記の属性クラスCの属性は一旦属性値が与えられると変化することはないので、図1のマネジャ12のキャッシュ17の管理情報C2とエージェント10の管理情報M2との一致は常に保たれる。属性クラスA, Bの属性(管理情報M0, M1)は夫々マネジャ12のキャッシュ15, 16に格納され、CMIPプロトコル・スタック18のコヒーレンシ・プロトコルによりコヒーレンシが保たれる。属性クラスDの属性(管理情報M3)はCMIPプロトコル・スタック18のM-GET, M-SETを用いて属性値の読み出し、変更が可能である。

【0026】図1に戻って説明するに、属性クラスCのMOについては、マネジャ12とエージェント10との間の通信が必要なのは初期値の設定の場合のみであるから、通常のアクセスにおいてはキャッシュ・コヒーレンシの維持のためのCMIPを必要としない。属性値の設定後読み出しが殆ど起こる可能性の無い属性クラスDについては、キャッシュを適用する必要性は少ないので、キャッシュを用いない。

【0027】マネジャ12において管理シーケンス14の実行時に管理情報M0, M1, M2に対してはそれぞれのキャッシュ15~17の管理情報C0, C1, C2をアクセスし、管理情報M3に対しては直接CMIPのコマンドを用いて属性値の制御を行うことにより、管理シーケンス14の実行を高速化することができる。管理

情報C0, C1に対してはそれぞれのコヒーレンシ要求に応じたプロトコルを適用する。すなわち、属性クラスA、属性クラスBについてはキャッシュと独立にエージェント10上MOの属性値が変化する可能性があり、コヒーレンシ・プロトコルにより両者の一致をとる必要がある。MOの属性値が変化する理由として、エージェント10の自立的な動作による変更、あるいはまた複数のマネジャが通信網内に存在する場合に、他のマネジャによるM-SETの結果変化するなどが考えられる。

【0028】次にコンパイル時のインタフェース処理部プログラムの自動生成について説明する。図3はTMN通信網管理制御に適用した場合のシステムフローチャートを示す。ここで、管理情報M0~M3, C0~C2は図1と対応している。マネジャ12上で実行される管理シーケンス14はC++のような高級言語とIDLのようなインタフェース記述言語を用いて記述される。一方、CMIPによるMOへのアクセスのインタフェースはGDMO (Guideline for the Definition of Managed Object) により規定されており、IDLコンパイラ22は通信網管理シーケンスからアクセスされるMOのGDMO記述20を参照しながら、マネジャ12上のインタフェース処理部プログラム(インタフェース処理手段)であるstub.c、Manager 起動時の初期化プログラム(初期化手段)であるmanager\_init.c、エージェント上のインタフェース処理部プログラム(インタフェース処理手段)であるskeleton.c Agent起動時の初期化プログラム(初期化手段)であるagent\_init.cを自動的に生成する。

【0029】GDMO記述による図2のようなMO記述において通常与えられるGET, GET-REPLACEのような属性分類のみではコヒーレンシ要求度に関して十分な情報を与えないので、図5に示す如くコメント文の中に、pragma文でIDLコンパイラ22への指示を埋め込むことにより実現する。IDLコンパイラ22は”-”で始まるコメント文の中にpragma文を見つけると、その指示に従ってマネジャ側においてキャッシュ領域の確保、コヒーレンシ・プロトコルの選択を行い、以後のキャッシュへのアクセス方法を決定する。具体的にはキャッシュ領域の確保、初期値の設定を行うプログラムをmanager\_init.cに出力し、キャッシュへのアクセスを行うプログラムをstub.cに出力する。同じ情報を用いてエージェント10側で使用するプログラムについても、それぞれの属性クラスについて対応するプロトコルを用いるので、同様に初期化プログラムagent\_init.c、インタフェース処理部skeleton.cを出力する。

【0030】このようなpragma文による属性クラスの指定は、通信管理シーケンスの移植性、及びキャッシュを使用しない通常のIDLコンパイラ22による通信網管理シーケンスのコンパイルに何らの害も及ぼさない。理由としてまず通信網管理シーケンスを実行するプログラ

ムから見た場合、キャッシュの存在は透明であり意識する必要がなく、プログラムの実行結果はキャッシュの有無に関わらない。またIDLコンパイラ22から見た場合、pragma文を理解しないコンパイラにとっては単なるコメント文と同じであり、キャッシュを使用することなく通常のCMIPの使用によって通信網管理シーケンスを実現するからである。

【0031】図4はIDLコンパイラ22の処理構成図を示す。同図中、IDL文法解析部25は通信網管理シーケンス14内のインタフェース部IDL記述14aの解析を行って属性クラス別コード要素生成部26に供給する。また、GDMO文法解析部27はGDMO記述20を解析し、この解析結果を用いて属性要素の属性クラス弁別部28で属性クラスを弁別し属性クラス別コード要素生成部26に供給する。

【0032】属性クラス別コード要素生成部26ではIDL解析結果及び属性クラス弁別結果に基づいて属性クラス別コード要素を生成する。マネジャ側コード要素生成部30は上記属性クラス別コード要素からマネジャで使用するプログラムstub.c及びmanager\_init.c31を生成し、エージェント側コード要素生成部2は上記属性クラス別コード要素からエージェントで使用するプログラムskeleton.c及びagent\_init.cを生成する。

【0033】ここで、図5に示すpragma文を補ったMO定義と、これに対応して図6に示すIDL定義が与えられた場合について説明する。図6のステップS520に用いられている\_objectInstance、\_ObjectIdなどのタイプのデータ構造の定義は、ITU-Tのドキュメントで定義されている。IDLコンパイラ22は、図6のIDと定義の文法解析を行い、各属性に対応して以下のようなコード要素をstub.c及びskeleton.cの一部として生成する。例えば、図5のステップS260の属性crossConnectionIdに対してはGET-ONLY属性であるので、

```
ObjectId crossConnection_get_crossConnectionId();
```

という関数を生成し、一方ステップS270の属性administrativeStateに対しては

```
enum crossConnection_get_administrativeState();
void crossConnection_set_administrativeState();
```

という関数を生成する。通常のCMIPを用いるシーケンスにおいては、マネジャ12からの管理シーケンス中において属性administrativeStateの値を参照する場合、

```
adminState= crossConnection_get_administrativeState(o, ev);
```

により得ることができ、一方属性の値を設定する場合 crossConnection\_set\_administrativeState(o, ev, adminStateValue)

のように行う。マネジャ12側で起動されるそれぞれの

関数はstub.cに含まれ、IDLコンパイラ22によって生成されるコード要素である。ここでoはエージェント上のMOを参照するポインタであり、evはマネジャプロセスの環境変数を表わすポインタである。エージェント側ではマネジャ側で起動された crossConnection\_get\_administrativeState(), crossConnection\_set\_administrativeState() に対応して同名の関数を起動し、MOの実際の属性値の参照及び更新を行う。ここで、エージェント上で起動されるそれぞれの関数はskeleton.cに含まれ、IDLコンパイラ22によって生成されるコード要素である。

【0034】次に、キャッシュ管理について説明する。まず、キャッシュ領域の確保については、本発明の方式ではキャッシュ領域は通信網管理シーケンス14を実行するプログラムのデータ領域にとられるので、通常のパーソナル・コンピュータ、ワークステーションのオペレーティング・システム(OS)において容易に実現され、何ら特殊なハードウェアないしOSを必要としない。また、キャッシュ領域の実現にあたり、キャッシュされる各MOの属性とメモリ上の領域の対応があればどのような形態でも良く、通常のメモリを用いて容易に実現される。

【0035】すなわち、具体的には通信網管理シーケンス14からのM-CREATE要求によってエージェント10上にMOインスタンスが生成されるのと並行してマネジャ12上キャッシュ内にMO属性対応のキャッシュが確保される。又はM-GET/M-SET要求によってマネジャ側からエージェントへのMOへのアクセス要求が生じた時点でキャッシュが確保され、同じく通信網管理シーケンスからのM-DELETE要求によってエージェント10上MOインスタンスが消去されるのと並行してキャッシュ内のMO属性対応部分が消去され、あるいはキャッシュ内データが使用されなくなるとキャッシュ管理シーケンスによってLRU (Least Recently Used)などのアルゴリズムを用いて属性対応のキャッシュ内領域は解放される。

【0036】図7は、図5において示したcrossConnect MO属性のメモリ上のキャッシュ実現例を示している。

図7において、crossConnect\_Cache(o)は当該MOのキャッシュ領域を指すポインタである。ここで、oは、MOを参照するためのポインタであり、関数crossConnect\_Cache() はハッシュ法などの方法によって当該MOに対応するキャッシュ中の領域を計算する関数である。また図中 crossConnectionId, administrativeState などの領域はそれぞれの属性値を格納し、AdministrativeClass, C, AdministrativeClass, Bなどの領域は、それぞれの属性クラスを示している。またそれぞれの属性値は図中に示される属性番号〔1〕-〔7〕によって一意に指定することができる。属性クラス値は使用すべきコーデシ・プロトコルを指定しており、通信網管理シー

ケンス 1 4 から属性値の更新要求が起こると、マネジャ機能を実現している通信網管理システムは属性クラス値に従って適切なコヒーレンシ・プロトコルを起動することによりエージェント 1 0 上の MO 属性値とキャッシュ上の内容の一致を保つ。すなわち、キャッシュの存在は通信網管理シーケンス 1 4 の使用者からは透明になっている。

【0 0 3 7】また、図 8 はメモリ上のキャッシュの異なる実現例である。属性番号を付された各属性はポインタを介して間接的にアクセスされることにより、図 7 とは異なり、各属性を表わすデータはメモリ上の任意の場所に置くことが可能であり、またマネジャ 1 2 側で使用しない属性（〔3〕，〔4〕）についてはキャッシュしない、という選択が可能となる。

【0 0 3 8】ここで、各属性クラスで使用するコヒーレンシ・プロトコルについて説明する。属性クラス A では、任意の二つのキャッシュの内容が一致するまでの時間がある一定値（ $\tau$ ）以下となるように制御し、具体的には定期的（ $\tau$  以下）にエージェント側からマネジャ 1 2 側のキャッシュ 1 5 へブロードキャスト（Broadcast）を行い、キャッシュ 1 5 に置かれている属性値を更新する。このコヒーレンシ・プロトコルをプロトコル（A）と呼ぶ。

【0 0 3 9】属性クラス B に対しては厳密なコヒーレンシを確保する必要があるため、キャッシュ 1 6 を更新するためにクロックを用いる標準的なコヒーレンシ・プロトコルを適用する。この時、エージェント側からマネジャ側に、MO の属性値の変化を通知する必要があるため、CMIP における M-Event-Report のような機能を利用することができる。このコヒーレンシ・プロトコルをプロトコル（B）と呼ぶ。

【0 0 4 0】属性クラス C に対しては、キャッシュの内容の一致は常に保たれているので、Manager-Agent 間の通信は必要なく、キャッシュの値を参照するのみで良い。次にキャッシュ 1 5 ～ 1 7 の初期設定について説明する。マネジャ 1 2 とエージェント 1 0 との間の CMIP のアクセスを最小限にするために、コンパイル時に知られている情報を用いて可能な限り、manager \_\_init.c においてキャッシュ 1 5 ～ 1 7 を初期化する。

【0 0 4 1】まず、属性クラスで分割したキャッシュである属性分割キャッシュを使用し、かつ属性分割キャッシュを使用しない通常の CMIP に従って動作する網管理装置と相互運用性（Interoperability）を確保するためには、マネジャ側、エージェント側において適切な初期化操作が必要であり、IDL コンパイラ 2 2 はそのような初期化ルーチンをそれぞれ manager \_\_init.c, agent \_\_init.c として出力する。

【0 0 4 2】図 9 はマネジャ 1 2 の初期化ルーチンのフローチャートを示す。同図中、まずステップでエージェント側に属性分割キャッシュ及び属性分割更新プロトコ

ルを備えるか否かを問い合わせ、ステップ S 7 1 0 でエージェント側からの上記問い合わせ結果を判別する。問い合わせ結果が YES の場合はステップ S 7 2 0 に進んで属性分割キャッシュ及びプロトコル（A），（B）等の属性分割更新プロトコルの適用を行い、問い合わせ結果が NO の場合はステップ S 7 3 0 に進んで通常の CMIP を用いる属性更新を行う。

【0 0 4 3】図 1 0 はエージェント 1 0 の初期化ルーチンのフローチャートを示す。同図中、ステップ S 7 5 0 ではマネジャ側からの属性分割キャッシュ及び属性分割更新プロトコルを備えるか否かの問い合わせを受け取り、ステップ S 7 6 0 でキャッシュの使用状態を表現する Cache MO の属性を調べることで属性分割キャッシュが使用可か否かをチェックする。

【0 0 4 4】次にステップ S 7 7 0 で上記のチェック結果を問い合わせを行ったマネジャに送り返すと共に、ステップ S 7 8 0 で上記チェック結果を判別し、属性分割キャッシュが使用可の場合にはステップ S 7 9 0 でエージェント側からのブロードキャスト送信を可能とするためのイベントスケジューラ（event \_\_scheduler）の初期化を行う。これは属性値変更時においては、プロトコル（A），（B）のいずれにおいてもエージェント側から MO の属性値変化をマネジャ側にブロードキャスト送信する必要があり、それを可能とするためにエージェント 1 0 のプロセスの event \_\_scheduler を初期化し、CMIP の EventNotification の機能を用いて、属性値の変化をマネジャ側に通知するようにしている。

【0 0 4 5】次にキャッシュのアクセスについて説明する。図 1 1 はマネジャ 1 2 が実行するキャッシュアクセスのフローチャートを示す。プロトコル（A）の場合は GET 操作及び SET 操作において、プロトコル（B）の場合は GET 操作において、マネジャからキャッシュをアクセスすることになる。

【0 0 4 6】同図中、まずステップ S 8 0 0 でハッシュ関数などを用いて、対応する MO がキャッシュ内に存在するか否かを確認する。対応する MO がキャッシュ内に有ればステップ S 8 1 0 でキャッシュ内の対応する MO にアクセスし、無ければステップ S 8 2 0 でキャッシュ管理シーケンスを起動し、CMIP プロトコルによって対応する MO をキャッシュに書き込んでキャッシュ内容を更新する。この後、ステップ S 8 3 0 においてキャッシュ夫々の属性クラスに応じたコヒーレンシ・プロトコルつまりプロトコル（A），（B）を起動することによりキャッシュコヒーレンシを確保する。

【0 0 4 7】次にキャッシュ領域の管理について説明する。キャッシュはマネジャ 1 2 のメモリ上に確保されるが、メモリ領域に制限がある場合、アクセスする可能性があるすべての MO 属性をキャッシュにおくのは困難であるため、LRU、CLOCK などのメモリ管理アルゴリズムの適用により、キャッシュの有効利用をはかる。

具体的には、通信網管理シーケンスからのM-SETないしM-GET要求に対して、要求先のMOに対するキャッシュ内領域へのポインタ（図7におけるcrossConnect\_Cache）が存在すれば当該MOのキャッシュは存在し、ポインタがNULLならば当該MOのキャッシュは存在しない。当該キャッシュが存在しない場合、マネージャ機能を実現する通信網管理システムは属性クラスDに準じ、CMIPコマンドを直接用いてMO属性値のGET/SETを行う。

【0048】一例としてLRUメモリ管理が適用されている場合、キャッシュ領域が満杯になると、キャッシュ内のMOの属性のうちアクセスの履歴が古いものからキャッシュを消去し、対応するキャッシュ内領域へのポインタをNULLにリセットし、領域を確保する。新しく解放された領域に、新たにアクセスされたMO属性のキャッシュを確保する。

【0049】図12にLRUを用いた場合のキャッシュ管理シーケンスを示す。図中の番号はシーケンスの実行順序に対応する。まず、「0」として、マネージャ12がアクセスしようとするMO属性がキャッシュ15、16中に存在するか調べ、その応答「1」がNOである場合、Cache中に存在する属性データのうち、最近もっとも使われていない属性A2の領域を解放する。並行してエージェント10側へM-GETを用いて当該MO属性値を取りに行き「2」、MO属性値を得、解放された領域に格納する「3」。エージェント10側では、プロトコル（A）、（B）においてキャッシュ中のMO属性値の更新をBroadcastに行うため、どのマネージャがキャッシュ中にMO属性を持っているか知る必要がある。このため、マネージャ12はエージェント10側に対して、更新されたキャッシュ状態についてエージェント10側に通知し「4」、それに従ってエージェント10側ではキャッシュ状態を示すテーブル（Cache Status）を更新する「5」。Cache Statusは図中に示されるように、キャッシュ可能な属性（A1、A2）とそれらの属性をキャッシュしているマネージャの組を表わすテーブルであり、ここでMはマネージャのIDを表わす。図中に示した管理シーケンスの結果、属性A1については0からMへ変化し、属性A2についてはMから0へ変化する。

【0050】次に、エージェントの障害復旧について、図13を用いて説明する。同図中、マネージャ12はエージェント10のMO40の属性値をキャッシュ15～17に持っており、時刻t。以前でMOとキャッシュとの属性値はgで一致しているものとする。

【0051】また、エージェント10はCMIPを用いて通信を行っているマネージャ12に関するアソシエーション情報を不揮発性メモリ（不揮発性記録手段）41に記録している。このアソシエーション情報は図14に示す如く、その時点で自エージェント10と通信しているマネージャの名前（マネージャID）M1、M2、M3が列

挙されている。

【0052】時刻t。において、エージェント10が障害を起こし、MO40の属性値が不定（X）になったとする。時刻t。においてエージェント10が復帰したとすると、エージェント10はMO属性値の障害前の値に復帰するために不揮発性メモリ41を検索し、キャッシュを持っている可能性のあるマネージャM1、M2、M3にCMIPを用いて問い合わせ（R）を行い、各マネージャは問い合わせのあったMO属性に対応するキャッシュ15～17を持っている場合にそのキャッシュの値

（A）をエージェント10に返す。これにより、時刻t。以後はエージェント10上のMO40の属性値が障害以前の前の値（g）に復帰する。以後は通常のコヒーレンシ・プロトコルに基づいて動作する。

【0053】図15にエージェント10の障害復旧のフローチャートを示す。エージェントに障害が発生し、障害が回復した場合にこの処理を開始する。まず、ステップS900で不揮発性メモリ41に名前が記録されているマネージャに対してMO属性値の問い合わせを行う。また、エージェント10に不揮発性バックアップ装置が接続されていれば、これについてもMO属性値を問い合わせる。次にステップS910で問い合わせを行ったマネージャ及び不揮発性バックアップ装置からの応答を受信する。ステップS920では同一のMO属性についての複数の応答があった場合、そのMO属性の更新時間、即ちコミット時間が最も新しいものを採用する。これは、プロトコル（A）を適用する属性の場合、各マネージャのキャッシュの属性値が一時的に不一致の場合があるからである。ステップS930では採用した応答によってMO40を再現し、エージェント動作を復旧する。

【0054】すなわち、マネージャ上のキャッシュを障害時のバックアップとして利用することにより、特別なメカニズム設けることなくネットワークの耐障害性、信頼性を高めることができる。なお、エージェント10ではなく、マネージャが障害を起こした場合も同様であり、この場合は通常のmanager\_init.cにより自動的にコヒーレンシ・プロトコルが働きキャッシュの内容が障害以前の値に復帰するので、この場合は問題でない。

【0055】次に、本発明を階層的ネットワークに適用した場合について説明する。図16に階層的ネットワークの構成図を示す。同図中、ネットワーク50は複数のドメインに分割され、各ドメインがサブネットワーク51、52を構成している。NMはネットワーク50のマネージャ、SNM0、SNM1はサブネットワーク51、52夫々のマネージャを表わし、S、T夫々はネットワーク終端点（TTP）のコネクト装置であり、O、P、U、Vはサブネットワーク51のドメイン境界の終端点（CTP）のコネクト装置である。

【0056】ここで、ネットワークの終端点S、T間のパス設定は、次の順に実行される。



## (1) パス探索

(2) パス情報を表すTrail (伝送路)、CTP、TTP等のMO生成

## (3) パス導通 (Activate)

(1) のパス探索ではコスト的に最適なパスを選ぶためNMはSNM内のパスに関するコスト情報が必要になり(O-P, Q-Rの区間)、(2) で生成するMOの一部については、管理上NMとSNM両方からアクセスされる。例えばO, P, Q, Rなどのドメイン間境界のクロスコネクト装置のMOなどはその例である。この図においてNM, SNM0, SNM1がマネジャでありO, P, Q, Rがエージェントである。

【0057】この場合、MOの属性をコヒーレンシ要求に従って分割し、それぞれのクラスに異なるコヒーレンシ・プロトコルを適用する。図17はSNMによる属性更新シーケンスを表わしている。このようなシーケンスは、SNMがその管理ドメインであるサブ・ネットワークに関する管理シーケンスを実行するに当り、そのドメイン間境界にあるクロスコネクト装置のMO属性値の変更を行う場合に必要となる。

【0058】図16の例ではSNM0がO, P, U, Vの属性値変更を行う場合、SNM11がQ, R, W, Xの属性値変更を行う場合がこれに当る。また、図17では属性クラスBについてキャッシュ更新が成功する場合のみを示しているが、実際においては複数のマネジャがキャッシュを持つ場合キャッシュ更新が常に成功するとは限らない。とくに、ロック要求は複数のマネジャが同時にエージェントへ書き込みを行うことを防ぐために必要であるが、ロックが得られない場合シーケンスは失敗する。失敗の場合はシーケンスを途中で中断し、最初からやり直す。

【0059】ここで、属性クラスA クロスコネクトMO属性のうち、属性クラスAに類別したもの、パス探索におけるSNM内のパスに関するコスト情報を表わすコスト、テーブルなど。属性クラスB クロスコネクトMO属性のうち、属性クラスBに類別したもの。

【0060】図17 (A), (B) の例では、NM、エージェント、SNM間のメッセージの交換を、それぞれ属性クラスAと属性クラスBに分けて示している。ここでエージェントはCMIPにおけるエージェントを意味し、具体的にはQ~R, U~Xなどのクロスコネクト装置を表わしている。すなわち、MO及びその属性値はそれぞれエージェント上にあり、NM及びSNMはそのMO属性対応のキャッシュをメモリ上に持っている場合を示している。

【0061】シーケンス開始時(時刻 $t_0$ 以前)のMO属性値を $g$ とし、SNMにおいて実行中の管理シーケンスから時刻 $t_1$ にMOの属性値を $h$ とする書き込み

(V) が発生する場合を示している。図中NM, エージェント, SNMの柱内の値( $g, h, X$ )はそれぞれ各

時刻におけるNM上のMO属性対応のキャッシュ、エージェント上MOの属性値、SNM上のMO属性対応のキャッシュの値を示す。ここで、 $X$ はキャッシュが無効状態にあり、読み出し不可になっていることを表わす。以下、図中シーケンスのメッセージ種別、記号、及び時刻の意味について説明する。

【0062】 $g, h$  MOの属性値、あるいは対応するキャッシュの値。

$X$  キャッシュ無効状態。読み出しは不可。

10  $V$  SNMの管理シーケンスにおけるMO属性の書き込み要求。

$n\tau$  定期的にエージェント側からマネジャ側へブロードキャストが行われる時刻。

【0063】 $R$  SNMからエージェントへの属性値更新要求(Update Request)。

$L$  SNMからエージェントへのロック要求(Lock Request)。

$I$  エージェントからNMへのキャッシュ無効化要求(Invalid Request)。

20 【0064】 $A$  エージェントからSNMへの $L$ 又は $I$ メッセージに対応する返答(Acknowledgement)。

$B$  更新された新しい属性値の伝搬(ブロードキャスト)。

図17 (A) において、時刻 $t_1$ でSNMよりの更新要求エージェントに到着。時刻 $t_2$ で時間 $\tau$ 以下毎の定期的なエージェントからのブロードキャストによりNM, SNM内のキャッシュを更新する。時刻 $t_1 \sim t_2$ 間ではエージェント上のMOと、NM, SNM上のキャッシュとは不一致状態である。

30 【0065】図17 (B) において、時刻 $t_1$ で、エージェントからの $I$ メッセージによるNMのキャッシュ無効化が行われる。特に、エージェントが新しい値である $h$ を $B$ メッセージによってブロードキャストする場合、NM及びSNMにおいて全てのキャッシュが無効化されていることを確認する必要がある。すなわち、 $B$ メッセージの送出前にNM及びSNMから $A$ 及び $R$ メッセージが到着していなければならない。時刻 $t_1$ でNM, SNM夫々のキャッシュ更新が終了し、NM, SNM上のキャッシュ再び有効になる。時刻 $t_1 \sim t_2$ 間では、キャッシュ読み出し不可である。

【0066】図17 (A), (B) においては、属性クラスA及びBの場合のみ示しているが、属性クラスC, Dの場合はそれぞれ自明である。すなわち、属性クラスCの場合は定義より書き込みシーケンスは発生せず、属性クラスDの場合は通常行われるCMIPのM-SETコマンドを用いてマネジャはエージェント上MOの属性値を変更する。

【0067】このように、属性クラスAで必要とされるキャッシュ・コヒーレンシを維持するためのメッセージ数は属性クラスBにそれに比してはるかに少なく、また



属性クラスCの属性はコヒーレンシ維持の為のメッセージを必要としないことから、コヒーレンシ・プロトコルによるメッセージ数が1/10程度まで激減し、管理シーケンスの実行を高速化しながら、ネットワークが輻輳状態に陥る危険を排除することが可能となる。また、以上のコヒーレンシ・プロトコルは全てCMIPを用いて実現できるので、CMIPのプロトコル・スタックを守ったまま、すなわち現状の通信網管理システムを全く変更することなく本発明の通信網管理キャッシュ制御方式を導入することができる。

【0068】図18(A)、(B)はエージェントが時刻 $t_1$ に自発的に属性値を変更した場合の属性クラスA、B夫々のシーケンスを示している。図18(A)において、時刻 $t_1$ でエージェント上で属性値が $g$ から $h$ に変更。時刻 $t_2$ で定期的なエージェントからのブロードキャストにより、NM、SNM内のキャッシュを更新する。時刻 $t_1 \sim t_2$ の間ではエージェント上のMOと、NM、SNM上のキャッシュとは不一致状態。

【0069】図18(B)において、時刻 $t_1$ でエージェントからのIメッセージによるNM、SNMの無効化が行われる。この後、NM、SNMからエージェントへAメッセージが到着するとエージェントはNM、SNMへBメッセージをブロードキャストする。時刻 $t_2$ でNM、SNM夫々のキャッシュ更新が終了し、NM、SNM上のキャッシュが再び有効になる。

【0070】図19(A)、(B)はNMが時刻 $t_1$ に属性値を変更する場合の属性クラスA、B夫々のシーケンスを示している。図19(A)において、時刻 $t_1$ でNMよりの更新要求エージェントに到着。時刻 $t_2$ で時間 $\tau$ 以下毎の定期的なエージェントからのブロードキャストによりNM、SNM内のキャッシュを更新する。時刻 $t_1 \sim t_2$ の間ではエージェント上のMOと、NM、SNM上のキャッシュとは不一致状態である。

【0071】図19(B)において、時刻 $t_1$ で、エージェントからのIメッセージによるSNMのキャッシュ無効化が行われる。特に、エージェントが新しい値である $h$ をBメッセージによってブロードキャストする場合、NM及びSNMにおいて全てのキャッシュが無効化されていることを確認する必要がある。すなわち、Bメッセージの送出前にSNM及びNMからA及びRメッセージが到着していなければならない。時刻 $t_2$ でNM、SNM夫々のキャッシュ更新が終了し、NM、SNM上のキャッシュ再び有効になる。時刻 $t_1 \sim t_2$ の間では、キャッシュ読み出し不可である。

【0072】図20(A)、(B)はSNMにおいて実行中の管理シーケンスにおいて読み出し要求が発生した場合の属性クラスA、B夫々のシーケンスを示している。時刻 $t_1$ において発生する読み出し要求(V)に対し、SNM上のキャッシュの値( $g$ )が直ちに返される(W)。これは属性クラスA、B及びCに対して同じで

あり、何となればキャッシュ・コヒーレンシ・プロトコルはエージェント上のMO属性値とNM、SNM上のキャッシュの値が等しい( $g$ )ことを保証するからである。一方、属性クラスDについては、通常行われるCMIPのM-GETコマンドを用いてマネージャはエージェント上の属性値を読み出す。

【0073】図21(A)、(B)はNMにおいて実行中の管理シーケンスにおいて読み出し要求が発生した場合の属性クラスA、B夫々のシーケンスを示している。

10 時刻 $t_1$ において発生する読み出し要求(V)に対し、NM上のキャッシュの値( $g$ )が直ちに返される

(W)。これは属性クラスA、B及びCに対して同じであり、何となればキャッシュ・コヒーレンシ・プロトコルはエージェント上のMO属性値とNM、SNM上のキャッシュの値が等しい( $g$ )ことを保証するからである。一方、属性クラスDについては、通常行われるCMIPのM-GETコマンドを用いてマネージャはエージェント上の属性値を読み出す。

【0074】このように、属性分割を行い、かつキャッシュを使用することにより、使用しない場合に比較して読み出し時間を大幅に短縮し、同時に読み出しに伴うマネージャ・エージェント間のCMIPのメッセージ数を大きく削減する。このように、本発明によればキャッシュを用いて管理シーケンス実行の大幅な高速化を可能にしながら、コヒーレンシ・プロトコルによるメッセージ数を激減させることが可能となり、よって通信網管理の高速化に寄与する。また、キャッシュを障害時のバックアップとすることにより容易にかつ効率良く耐障害性を向上し、通信網の高信頼化に寄与するところが大きい。

30 【0075】

【発明の効果】上述の如く、請求項1に記載の発明によれば、管理主体にキャッシュを設けることにより、管理主体が実行する管理シーケンスを高速化することができ、管理情報の属性をキャッシュ・コヒーレンシ要求度に基づいてクラス分けすることによりキャッシュ管理の最適化をはかることが可能となる。

【0076】また、請求項2に記載の発明によれば、管理情報の定義から初期化手段が自動生成され、この初期化手段によって管理主体に属性分割キャッシュが確保され、かつ、その初期化が行われ、属性分割キャッシュを使用しない装置との相互運用性を確保できる。

【0077】また、請求項3に記載の発明によれば、管理情報の定義からインタフェース処理手段が自動生成され、このインタフェース処理手段によって属性クラス毎のコヒーレンシ・プロトコルが実行されるため、属性キャッシュ夫々に最適のコヒーレンシ・プロトコルが実行可能となる。

【0078】また、請求項4に記載の発明によれば、属性分割キャッシュが夫々に最適のコヒーレンシ・プロトコルを用いてキャッシュ・コヒーレンシが確保されるた

め、ネットワークの管理トラヒックの増大を防止してネットワークの負荷の増大を抑制できる。

【0079】また、請求項5に記載の発明によれば、属性分割キャッシュをリスト・リーセントリー・ユースト・アルゴリズムを用いて管理するため、キャッシュ領域が増大化すること避けることができ、メモリの有効利用が可能となる。また、請求項6に記載の発明によれば、不揮発性記録手段を有するため、代理主体と接続されている管理主体の情報を代理主体の障害時に確保できる。

【0080】また、請求項7に記載の発明によれば、代理主体の障害復旧時に、これと接続されていた管理主体のキャッシュから管理情報を復旧できるため、キャッシュを代理主体の障害復旧に有効利用でき、実用上きわめて有用である。

#### 【図面の簡単な説明】

【図1】本発明の構成図を示す。

【図2】MO定義を示す図である。

【図3】キャッシュ生成と初期値設定のシステムフローチャートである。

【図4】IDLコンパイラの処理構成図である。

【図5】MO定義を示す図である。

【図6】IDL定義を示す図である。

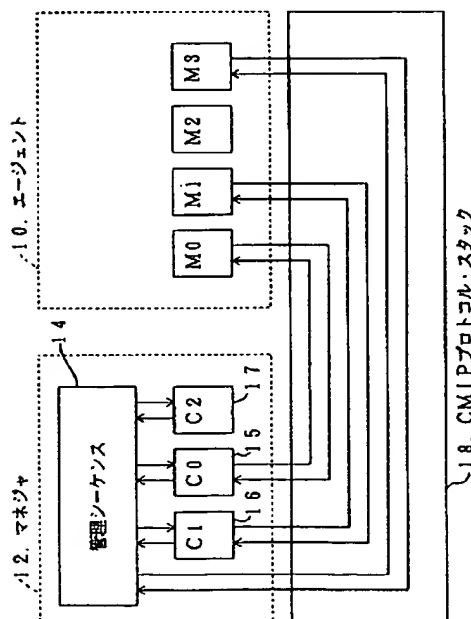
【図7】キャッシュ実現例を示す図である。

【図8】キャッシュ実現例を示す図である。

【図9】マネジャの初期化ルーチンのフローチャートである。

【図1】

本発明の構成図



【図10】エージェントの初期化ルーチンのフローチャートである。

【図11】マネジャのキャッシュアクセスのフローチャートである。

【図12】キャッシュ管理を説明するための図である。

【図13】エージェントの障害復帰を説明するための図である。

【図14】不揮発性メモリの記録形式を示す図である。

【図15】エージェントの障害復旧のフローチャートである。

【図16】階層的ネットワークの構成図である。

【図17】SNMの属性値書き込みシーケンス図である。

【図18】エージェントの属性値書き込みシーケンス図である。

【図19】NMの属性値書き込みシーケンス図である。

【図20】SNMの属性値読み出しシーケンス図である。

【図21】NMの属性値読み出しシーケンス図である。

#### 20 【符号の説明】

10 エージェント

12 マネジャ

15～17 キャッシュ

18 CMIPプロトコル・スタック

20 GDMO記述

41 不揮発性メモリ

【図6】

IDL定義を示す図

```

S510 /*
      type __ObjectInstance ' __ObjectId'
      is defined in X711 document.
      */
S550
      typedef __ObjectInstance ObjectInstance;
      typedef __ObjectId ObjectId;

      interface crossConnection : X721_top, MO_top {
S600         readonly attribute ObjectId crossConnctionId;
              attribute enum administrativeState;
              readonly attribute enum operationalState;
              readonly attribute enum signalType;
              readonly attribute ObjectInstance fromTermination;
S650         readonly attribute ObjectInstance toTermination;
              readonly attribute enum directionality;
      } :
  
```

【図 2】

MO定義を示す図

```

S 1 0 | crossConnection MANAGED OBJECT CLASS
      | DERIVED FROM "Recommendation X.721: 1992":top;
      | CHARACTERIZED BY crossConnectionPackage PACKAGE
      | BEHAVIOUR crossConnectionBehaviour ;
S 5 0 | ATTRIBUTES
      | crossConnectionId
      | GET;
      | "Recommendation X.721 : 1992":administrativeState
      | GET-REPLACE;
S 1 0 0 | "Recommendation X.721 : 1992":operationalState
      | GET;
      | signalType
      | GET;
      | fromTermination
S 1 5 0 | GET;
      | toTermination
      | GET;
      | directionality
      | GET::;
S 2 0 0 | REGISTERED AS {m31000objectClass 15 } ;

```

【図 5】

MO定義を示す図

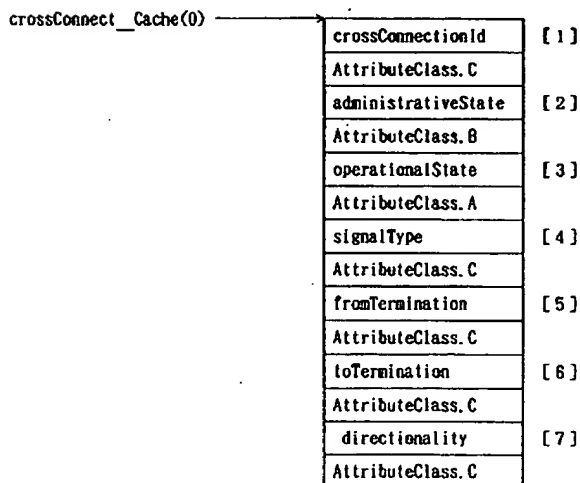
```

S 2 1 0 | crossConnection MANAGED OBJECT CLASS
      | DERIVED FROM "Recommendation X.721: 1992":top;
      | CHARACTERIZED BY crossConnectionPackage PACKAGE
      | BEHAVIOUR crossConnectionBehaviour ;
S 2 5 0 | ATTRIBUTES
      | crossConnectionId -- #pragma : AttributeClass.C
      | GET;
      | "Recommendation X.721 : 1992":administrativeState
      | -- #pragma : AttributeClass.B
      | GET-REPLACE;
S 3 0 0 | "Recommendation X.721 : 1992":operationalState
      | -- #pragma : AttributeClass.A
      | GET;
      | signalType -- #pragma : AttributeClass.C
      | GET;
S 3 5 0 | fromTermination -- #pragma : AttributeClass.C
      | GET;
      | toTermination -- #pragma : AttributeClass.C
      | GET;
S 4 0 0 | directionality -- #pragma : AttributeClass.C
      | GET::;
      | REGISTERED AS {m31000objectClass 15 } ;

```

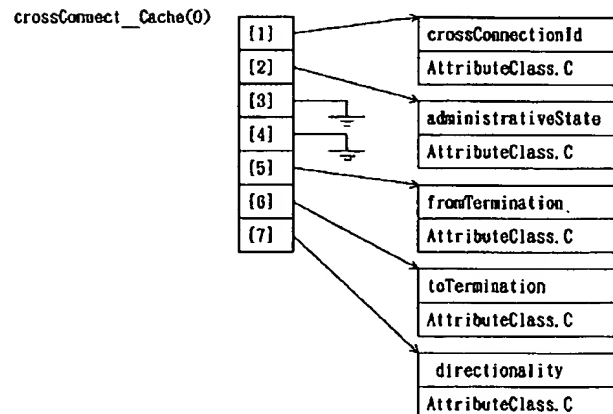
【図 7】

キャッシュ実現例を示す図



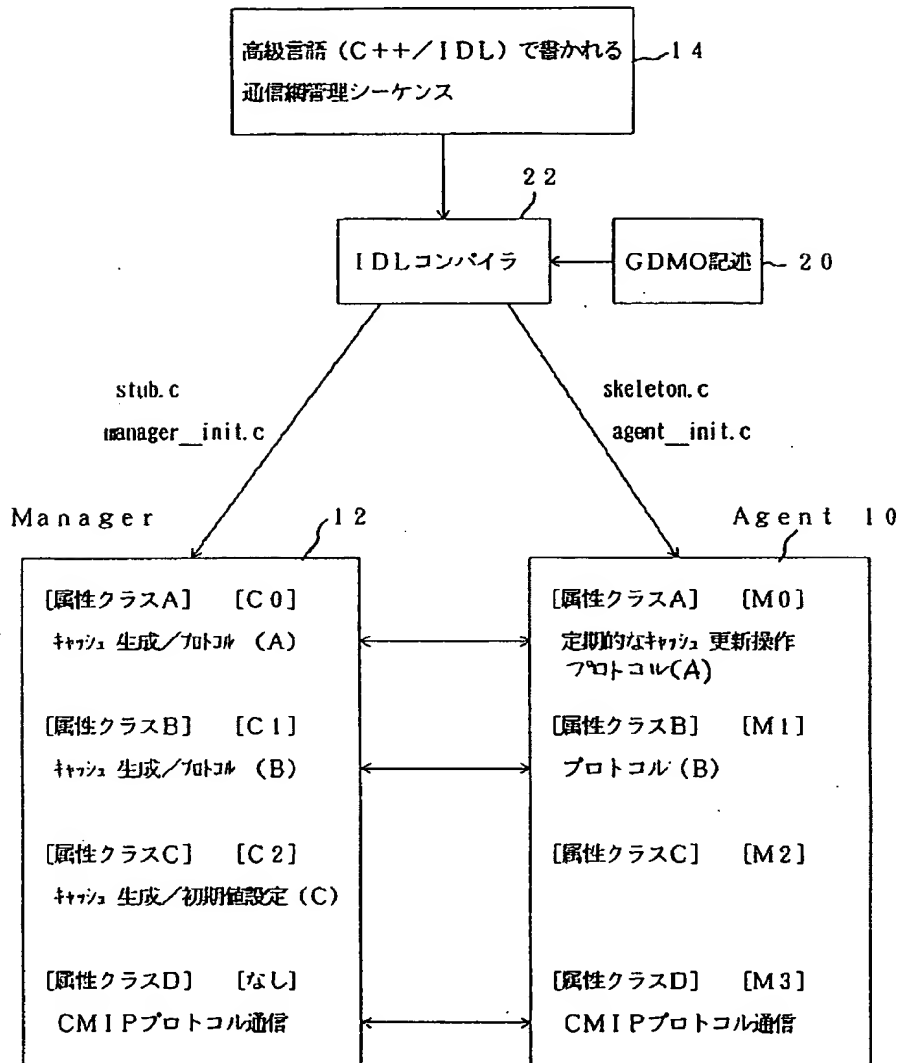
【図 8】

キャッシュ実現例を示す図



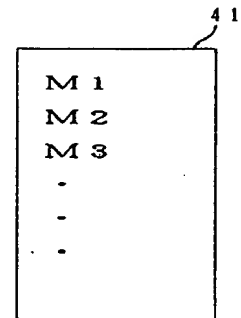
【図 3】

キャッシュ生成と初期設定のシステムフローチャート



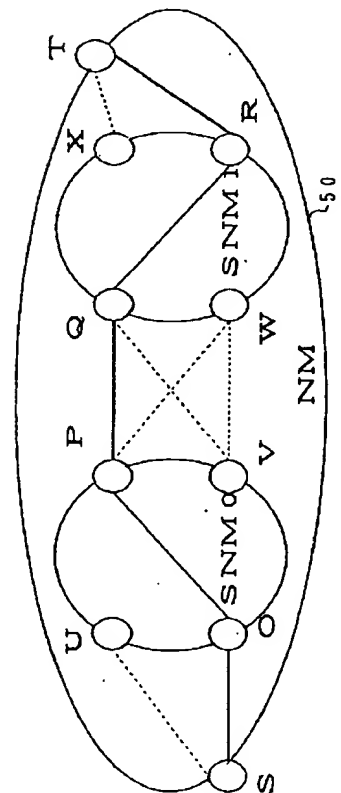
【図 1 4】

不揮発性メモリの記録形式を示す図



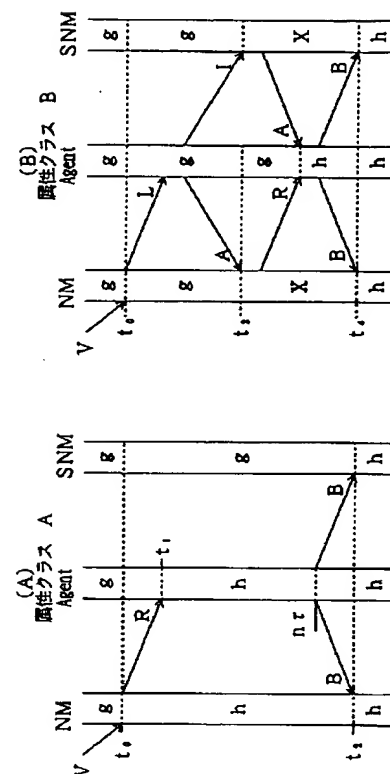
【図 1 6】

階層的ネットワークの構成図



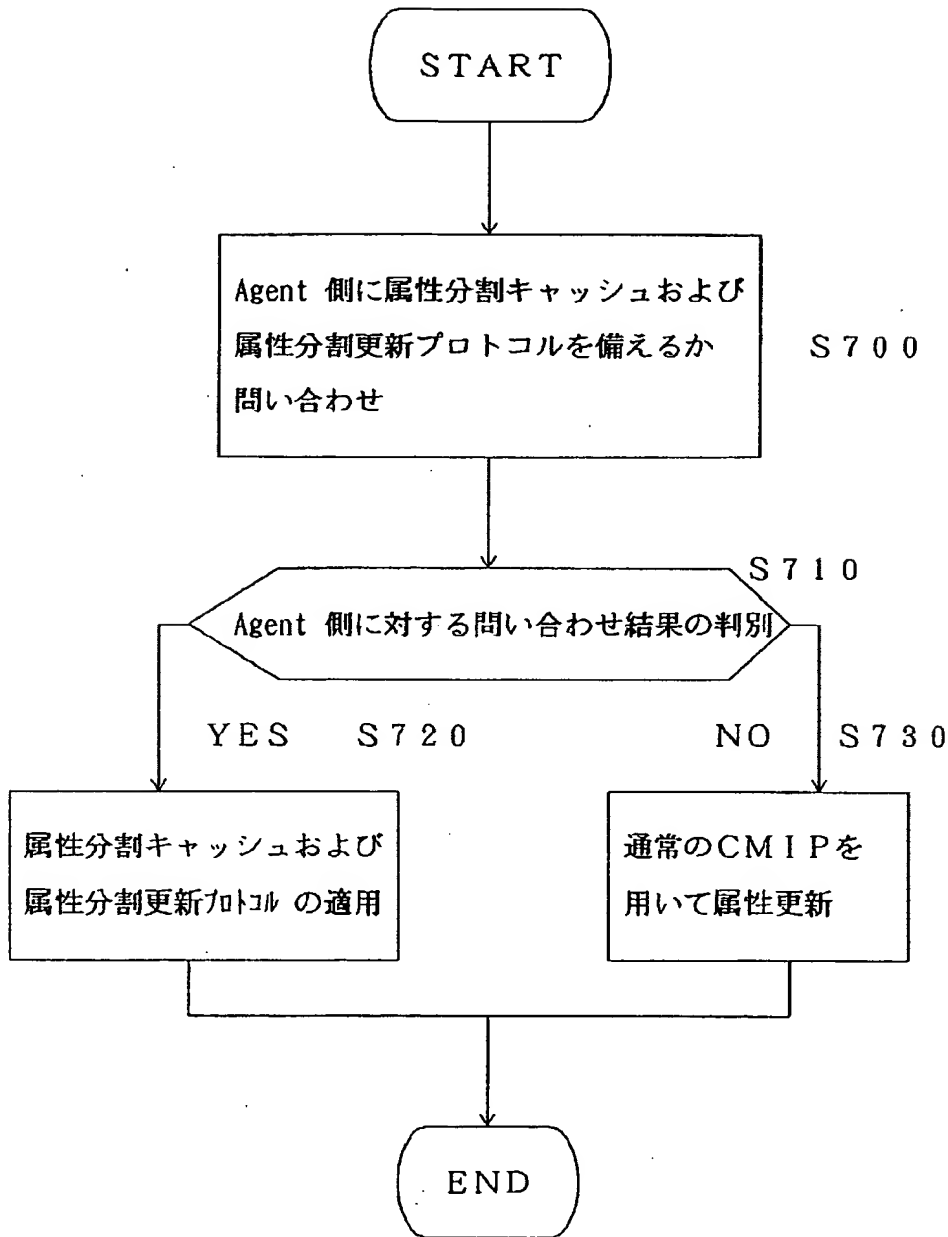
【図 19】

## NMの属性値書き込みシーケンス図



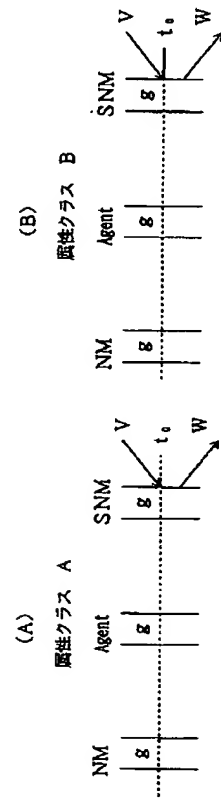
【図 9】

マネージャの初期化ルーチンのフローチャート



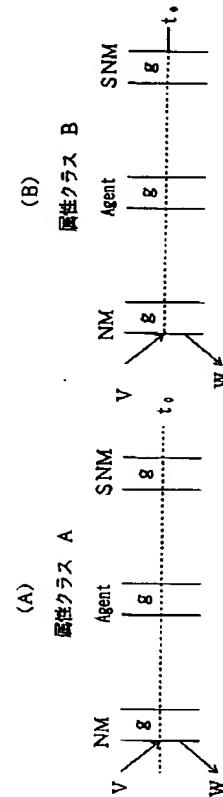
【図 20】

SNMの属性値読み出しシーケンス図



【图 2 1】

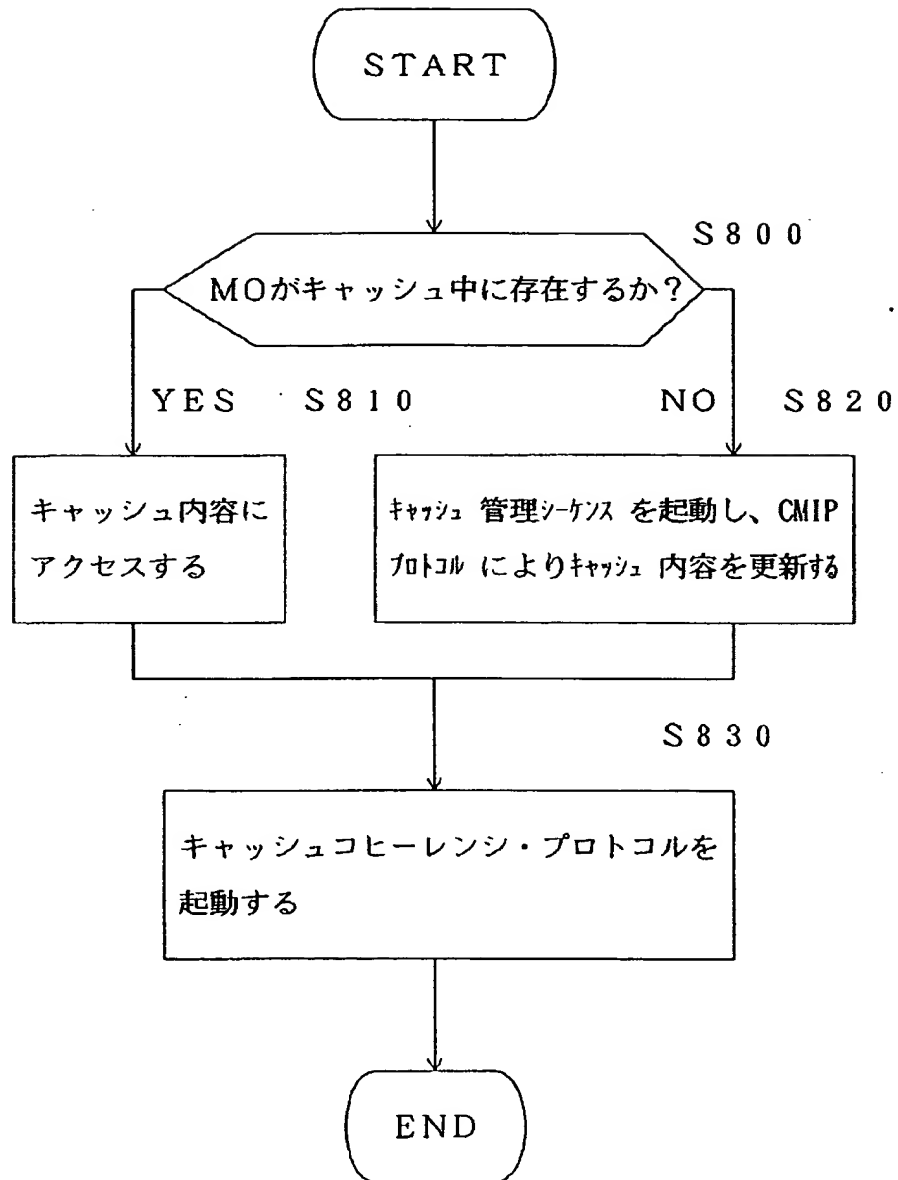
### NMの属性値読み出しシーケンス図





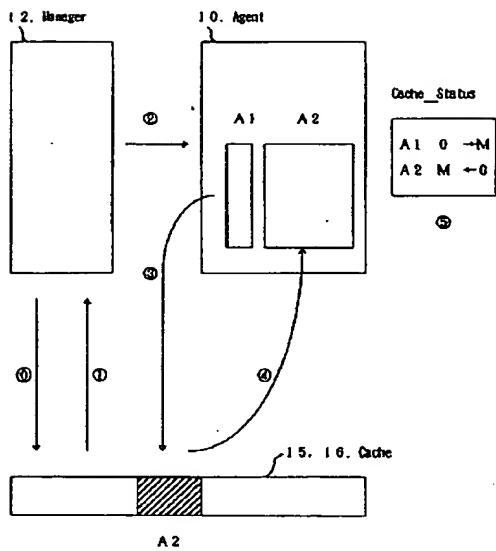
【図 1 1】

## マネジャのキャッシュアクセスのフローチャート



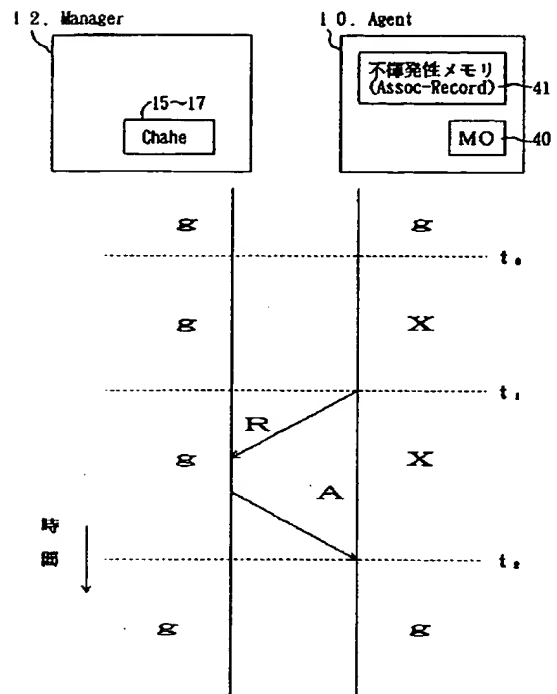
【図 12】

キャッシュ管理を説明するための図



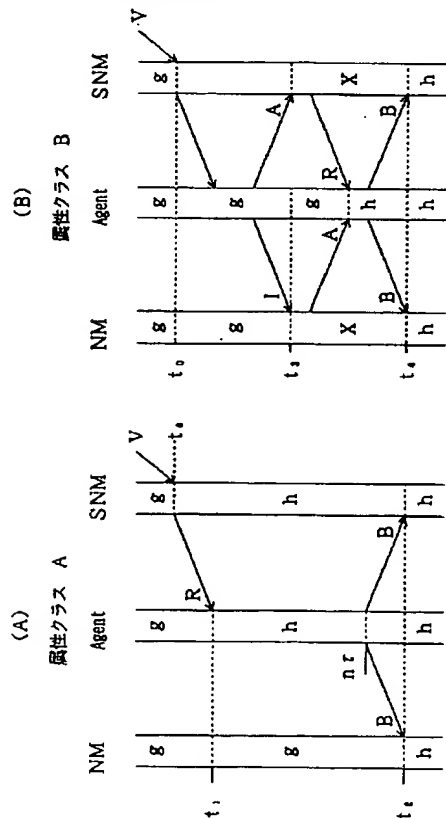
【図 13】

エージェントの障害復帰を説明するための図



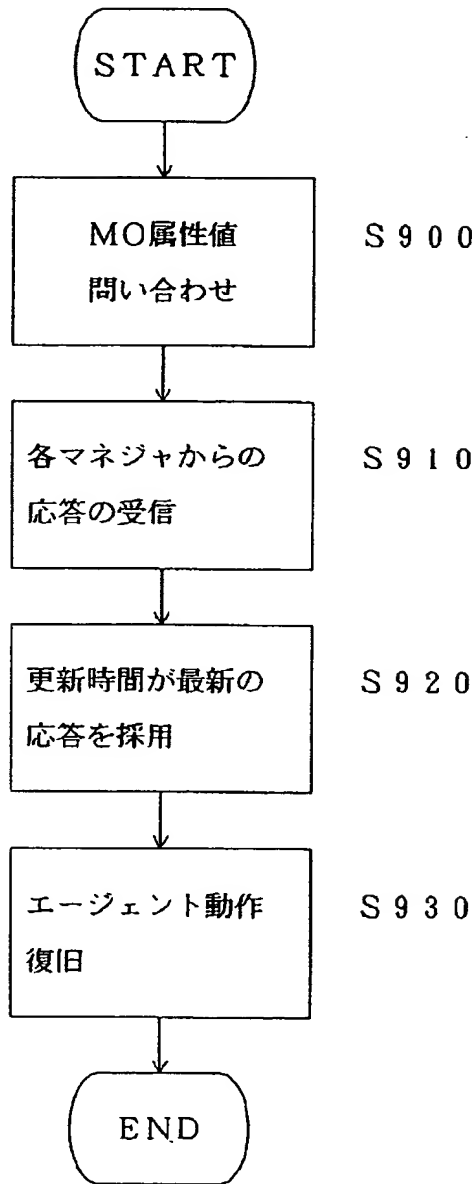
【図 17】

SNMの属性書き込みシーケンス図



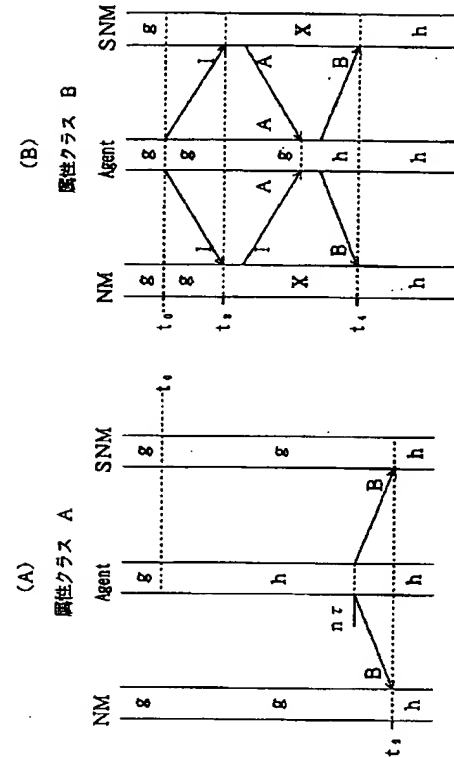
【図 15】

## エージェントの障害復旧のフローチャート



【図 18】

## エージェントの属性書き込みシーケンス図



フロントページの続き

(72)発明者 東 充宏  
神奈川県川崎市中原区上小田中1015番地  
富士通株式会社内

(72)発明者 中条 孝文  
神奈川県川崎市中原区上小田中1015番地  
富士通株式会社内